Collaboration in Distributed Systems by means of an Awareness-based Learning Model

*Mauricio Paletta[1] and Pilar Herrero[2]

[1] Centro de Investigación en Informática y Tecnología de la Computación (CITEC)
Universidad Nacional Experimental de Guayana (UNEG)
Av. Atlántico, Ciudad Guayana, 8050, Venezuela
mpaletta@uneg.edu.ve
[2] Facultad de Informática, Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo S/N; 28660; Madrid, Spain
pherrero@fi.upm.es

Phone number and Fax number of corresponding author:

Pho: +58 286 9239337
Fax: +58 286 9239717
Cel: +58 414 8950288

Abstract. Distributed systems have become increasingly common because they offer significant computational power and are cost-effective and scalable. Moreover, collaboration between users that are part of these distributed systems improves efficiency and effectiveness for a better utilization of this computational power. Due to this, new specific collaboration models for distributive systems are needed for enabling effective collaboration between users of these dynamic environments. This paper presents AMBAR, an awareness-based learning model for distributed environment with some patent literature related with learning and collaboration systems. AMBAR allows nodes to accomplish an effective collaboration by means of a multi-agent architecture in which agents are aware of its surroundings by means of a parametrical and flexible use of this information. This approach makes use of heuristic strategies to improve effectiveness and efficiency of collaboration in these environments. Results presented in this paper prove this assertion.

Keywords: Awareness, collaboration, distributed system, efficiency, effectiveness, learning, multi-agent system.

Short running Title: AMBAR-Model

1 Introduction

Most of the distributed system work toward providing reliable, customized and QoS guaranteed dynamic computing environments for end-users. The success of achieving this goal in proper time (efficiency) and/or to obtain the higher quality of results (effectiveness) in these dynamic and distributed environments depends on implementing an appropriate collaboration model between nodes in the system. Moreover, this collaboration mechanism should include learning abilities necessary for the use of the previous experience acquired (with situations that occurred in the past) in order to improve new collaborations required. Learning-based heuristic techniques seem to be a good alternative to achieve this goal.

On the other hand, according to CSCW (Computer Supported Cooperative Work) awareness is a useful concept used to achieve cooperation and collaboration in distributed environments because it increases communication opportunities [1]. A collaborative process is leaded by five processes [2, 3]: 1) co-presence, that gives the feeling that the user is in a shared environment with some other user at the same time; 2) awareness, a process where users recognize each other´s activities on the premise of co-presence; 3) communication; 4) collaboration which together with communication permit users to collaborate with each other to accomplish the tasks and common goals; and 5) coordination which is needed to resolve the conflicts towards effective collaborations.

In the same order of ideas, in CSCL (Computer Supported Collaborative Learning), awareness plays an important role as it promotes collaboration opportunities in a natural and efficient way [4], and it improves the effectiveness of collaborative learning. Related to this, Gutwin et al identified the following types of awareness [5]: social, task, concept, workspace, and knowledge.

By using a specific interpretation of the awareness concept, this paper presents AMBAR (Awareness-based learning Model for distriButive collAborative enviRonment), a new MAS-based learning collaboration model for distributed environments endowed with heuristic-based strategies aiming to take into account the information of awareness collaborations occurring in the environment for achieving the most appropriate future awareness situations.

Since a complete collaboration model for distributed environments has to do with different issues, this paper also includes the details of these elements associated with AMBAR and this includes: 1) the architecture used for designing the agents related with the model; 2) the awareness representation and the collaboration process associated with this representation; 3) the negotiation mechanism required to deal with saturated conditions; 4) the mutual exclusion strategy needed to synchronize the use of critical sections related with the share activities where agents in the system are immerse, and 5) the heuristic-based strategies, including those associated with learning abilities, defined for the model.

The remainder of this paper is organized as follows. Some background aspects are showed in section 2. Section 3 describes the AMBAR model proposed in this paper. Details of the implementation and the results of the evaluation of the model are showed in Section 4. Some related work is given in Section 5. Finally, the last section includes the conclusions and outgoing future research related to this work.

2 Technical backgrounds

Due to the fact that AMBAR uses heuristic techniques, this Section presents some background information related with the following aspects: 1) simulated annealing method; and 2) artificial neural networks.

2.1 Simulated Annealing Method

Simulated Annealing (SA) [6, 7] is a random-search technique (a generalization of a Monte Carlo method) that exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system. It forms the basis of an optimization technique to solve combinatorial and other problems. The concept originates from the way in which crystalline structures are brought to more ordered states by an "annealing process" of repeated heating and slowly cooling the structures.

In SA, a system is initialized at a temperature T with some configuration, whose energy is evaluated to be E. A new configuration/solution is constructed by applying a random change to the current configuration, and the change in energy dE is computed. The new solution is unconditionally accepted as long as if it lowers the energy of the system. However, if the energy of the system is increased by the change, the new configuration might be accepted depending on the probability distribution of Boltzmann [7]. This process is repeated sufficient times at the current temperature to sample the search space, then the temperature is decreased and the process is repeated at the successively lower temperatures until a frozen state is achieved.

This procedure allows the system to move to lower energy states while still jumping out of local minima (especially at higher temperatures) due to the probabilistic acceptance of some upward moves. SA theory states that if temperature is lowered sufficiently slowly (by carefully controlling the rate of cooling), the solid will reach thermal equilibrium, which is an optimal state (global optimum).

To solve a problem through this strategy it is necessary to define the following elements: 1) The representation of a valid solution to the problem; 2) the cost function (energy) that measures the quality of each solution; 3) the mechanism of transition of the space of solutions from t to t+1 (dynamic of the model); 4) parameters used to control the rate of cooling.

2.2 Artificial Neural Network (ANN)

ANNs are arrays of processing units (artificial neurons) that are interconnected with one another, forming the network. Any edge has a weight that makes the role of synapses in the neurophysiologic system. Through an appropriate adjustment of these synaptic weights, the ANNs are able to "learn" the association between a characteristic set of input (incentives) and output (response) vectors regarding a specific problem.

Moreover, the operation of an ANN is given by three basic properties: 1) a nonlinear transfer function associated with the processing unit; 2) topology interconnection units and 3) the learning or weights adjustment law (algorithm). The specific description of these elements makes an ANN model different from another. Neural-Gas (NGAS), Radial Based Function Network (RBFN) and Multi-Layer Perceptron (MLP) are three examples.

2.2.1 Vector quantization and neural-Gas

Vector Quantization (VQ) [8-12] is the process of quantizing n-dimensional input vectors to a limited set of n-dimensional output vectors referred to as *code-vectors*. The set of possible *code-vectors* is called the *codebook*. The *codebook* is usually generated by clustering a given set of training vectors (called *training set*). Clustering can be described then, as the process of organizing the *codebook* into groups whose members share similar features in some way. The goal of clustering is to reduce large amounts of raw data by categorizing it in smaller sets of similar items.

Neural-Gas (NGAS) [13] is a VQ technique with soft competition between the units. In each training step, the squared Euclidean distances between a randomly selected input vector x_i from the training set and all *code-vectors* m_k are computed; the vector of these distances, expressed in (1) is d. Each centre k is assigned a rank $r_k(d)$ = 0, …, N-1, where a rank of 0 indicates the closest and a rank of N-1 the most distant from the centre to x. The learning rule is expressed as it is indicated in (2).

$$d_{ik} = \|x_i - m_k\| = (x_i - m_k)^T (x_i - m_k) \quad (1)$$

$$m_k = m_k + \varepsilon * h_\rho[r_k(d)](x - m_k) \quad (2)$$

$$h_\rho(r) = e^{(-r/\rho)} \qquad (3)$$

A monotonically decreasing function of the ranking that adapts all the centers, with a factor exponentially decreasing with their rank is represented in (3). The width of this influence is determined by the neighborhood range ρ. The learning rule is also affected by a global learning rate ε. The values of ρ and ε decrease exponentially from an initial positive value ($\rho(0)$, $\varepsilon(0)$) to a smaller final positive value ($\rho(T)$, $\varepsilon(T)$) according to expressions (4) and (5) respectively, where t is the time step and T the total number of training steps, forcing more local changes with time.

$$\rho(t) = \rho(0)[\rho(T)/\rho(0)]^{(t/T)} \qquad (4)$$

$$\varepsilon(t) = \varepsilon(0)[\varepsilon(T)/\varepsilon(0)]^{(t/T)} \qquad (5)$$

2.2.2 Radial Based Function Network (RBFN)

Radial based functions were originally developed to discuss problems involving the adaptation of irregular topographic contours through a series of geographic data [14, 15]. ANN's based on this technique (RBFNs) are among the best choices in models out there as an alternative to achieve excellent results in alignment of data caused either by stochastic or deterministic functions [16].

This model consists of three layers of processing units: the input layer, the output layer and a layer of hidden units. Information flows in one direction, from the input layer (elements sensors) to a layer of units that shows the system output. On the way, the information is processed in part by the intermediate (hidden) units.

The learning process is a supervised-based strategy and provides a method to adjust the synaptic weights. Therefore the network learns $(\vec{y}, \vec{s})$ correspondence, being $\vec{y}$ the input vector and $\vec{s}$ the associated output vector. The basis of this algorithm is the method of gradient descent that is used to minimize a function of quality. The most commonly function used is mean-squared error, whose expression can be seen in (6), where: O - output units; s - desired outputs; i - units index for output layer; μ - learning patterns index.

$$\mathrm{E}(\vec{w}) = \frac{1}{2}\sum_{\mu i}(s_i^\mu - O_i^\mu)^2 \qquad (6)$$

Expressions (7) and (8) are used to calculate the spread of hidden units and output units respectively. Where: V - hidden layer unit; y - input unit; j - hidden units index; k - input units index; f - activation function; σ - standard deviation.

$$V_j = f\left(-\sum_k \frac{(y_k^\mu - w_{jk})^2}{2\sigma^2}\right) \qquad (7)$$

$$O_i = \sum_j W_{ij} V_j \qquad (8)$$

$$f(x) = \exp(x) \qquad (9)$$

According to (7-9) as well as the adjustment weights expressions described below, this model has its foundation in the Gaussian, hence the use of the standard deviation. Expressions for adjusting weights between the output layer and the hidden layer are shown in (10). Expressions (11) and (12) are used to adjust the weights and standard deviations between the hidden layer and the input layer.

$$\begin{aligned} \Delta W_{ij} &= \eta \sum_\mu (s_i^\mu - O_i^\mu)(\sum_j W_{ij} V_j) V_j^\mu \\ \Delta W_{ij} &= \eta \sum_\mu \delta_i^\mu V_j^\mu \,;\, \delta_i^\mu = (\sum_j W_{ij} V_j)(s_i^\mu - O_i^\mu) \end{aligned} \qquad (10)$$

$$\Delta w_{jk} = \lambda \sum_{\mu i} (s_i^\mu - O_i^\mu)(\sum_j W_{ij} V_j)\, W_{ij} \left(\frac{y_k - w_{jk}}{\sigma_{jk}{}^2}\right) f\left(-\sum_k \frac{(y_k^\mu - w_{jk})^2}{2\sigma_{jk}{}^2}\right) \qquad (11)$$

$$\Delta \sigma_{jk} = \gamma \sum_{\mu i} (s_i^\mu - O_i^\mu)(\sum_j W_{ij} V_j)\, W_{ij} \left(\frac{(y_k - w_{jk})^2}{\sigma^3}\right) f\left(-\sum_k \frac{(y_k^\mu - w_{jk})^2}{2\sigma^2}\right) \qquad (12)$$

Data consists of real values in the interval [0, 1]. Any data outside this range must be adequately normalized. η, λ and γ are the corresponding learning factors to each weight matrix.

2.2.3 Multi-Layer Perceptron (MLP)

MLP [17] is one of the most used neural models for implementing a variety of problems. It is a layer-based topology network in which information flows in one direction, from an input layer (sensing elements) to a layer of units that shows the system output. On the way, the information is partially processed by different intermediate layers (hidden) units. The number of units of each layer is arbitrary and the outputs of the units of any single layer are connected with all units of the next layer (there is no interconnection between the units in the same layer).

As occurs in RBFN learning in MLP is also supervised. The most popular learning algorithm is backpropagation [18]. Expression for calculating or estimating the spread of hidden units is observed in (13). Expression (14) shows the formula for the computation of output units. Most typical activation functions used are sigmoid and hyperbolic tangent (see both in (15)).

$$V_j = f(h_j^\mu) = f(\sum_k w_{jk} y_k^\mu) \tag{13}$$

$$O_i = f(\sum_j W_{ij} V_j) \tag{14}$$

$$f(x) = \frac{1}{1+\exp(-x)}; \qquad f(x) = \tanh(x) \tag{15}$$

The expressions used for adjusting the synaptic weights (for training) are caused by the gradient of the error term given in (6) in relation to the weights W_{ij} that represent the variables in this process. In this regard (16) shows the expression for the adjustment of the weights between output layer and the last hidden layer; (17) shows the expression for the adjustment of weights among other hidden layers until the input layer is reached.

$$\begin{aligned} \Delta W_{ij} &= -\eta \frac{\partial E}{\partial W_{ij}} = \eta \sum_\mu (s_i^\mu - O_i^\mu) f'(\sum_j W_{ij} V_j) V_j^\mu \\ \Delta W_{ij} &= \eta \sum_\mu \delta_i^\mu V_j^\mu \,; \delta_i^\mu = f'(\sum_j W_{ij} V_j)(s_i^\mu - O_i^\mu) \end{aligned} \tag{16}$$

$$\begin{aligned} \Delta W_{ij} &= -\eta \frac{\partial E}{\partial w_{jk}} = \eta \sum_{\mu i} \delta_i^\mu W_{ij} f'(\sum_k w_{jk} y_k^\mu) y_k^\mu \\ \Delta W_{ij} &= \eta \sum_\mu \delta_j^\mu y_k^\mu \,; \delta_j^\mu = f'(\sum_k w_{jk} y_k^\mu) \sum_i \delta_i^\mu W_{ij} \end{aligned} \tag{17}$$

Data consists of real values either in the interval [0, 1] or [-1, 1] (depending of the activation function). η is the learning adjustment factor.

3 AMBAR: An Awareness-based Learning model for Collaborative Distributive Systems

This section presents the details of the AMBAR model by taking into account the following issues: 1) the architecture used for designing the agents related with the model; 2) the awareness representation and the collaboration process associated with this representation; 3) the negotiation mechanism required to deal with saturated conditions; 4) the mutual exclusion strategy needed to synchronize the use of critical sections related with the share activities where agents in the distributive system are immerse, and 5) the heuristic-based strategies, including those associated with learning abilities, defined for the model.

3.1 Agent architecture

To implement this multi-agent system, SOFIA (SOA-based Framework for Intelligent Agents) [19, 20] an architecture system was used. SOFIA focuses on the design of a common framework for Intelligent Agents with the following characteristics: 1) it merges interdisciplinary theories, methods and approaches, 2) it is extensible and open as to be completed with new requirements and necessities, and 3) it highlights the agent´s learning processes within the environment. SOFIA's general architecture contains four main components (see Fig. (**1**)):

1) The Embodied Agent (IA-EA) or the "body": It is a FIPA based structure [21] because it has a Service Directory element which provides a location where specific and correspondent services descriptions can be registered. The IA-EA encloses the set of services related to the abilities of sensing stimuli from the environment and interacting with it.

2) The Rational Agent (IA-RA) or the "brain": This component represents the agent's intelligent part and therefore, it encloses the set of services used by the agent to implement the process associated with these abilities. It is also a FIPA based structure.

3) The Integrative/Facilitator Agent (IA-FA) or the "facilitator": It plays the role of simplifying the inclusion of new services into the system as well as the execution of each of them when it is needed. The basic function of the IA-FA is to coordinate the integration between the IA-SV and the rest of the IA components. This integration is needed when a new service is integrated with the IA and therefore it is registered into the corresponding Service Directory, even when an existing service is executed.

4) The IA Services or "abilities" (IA-SV): It is a collection of individual and independent software components integrated to the system (the IA) which implements any specific ability either to the IA-EA or the IA-RA.

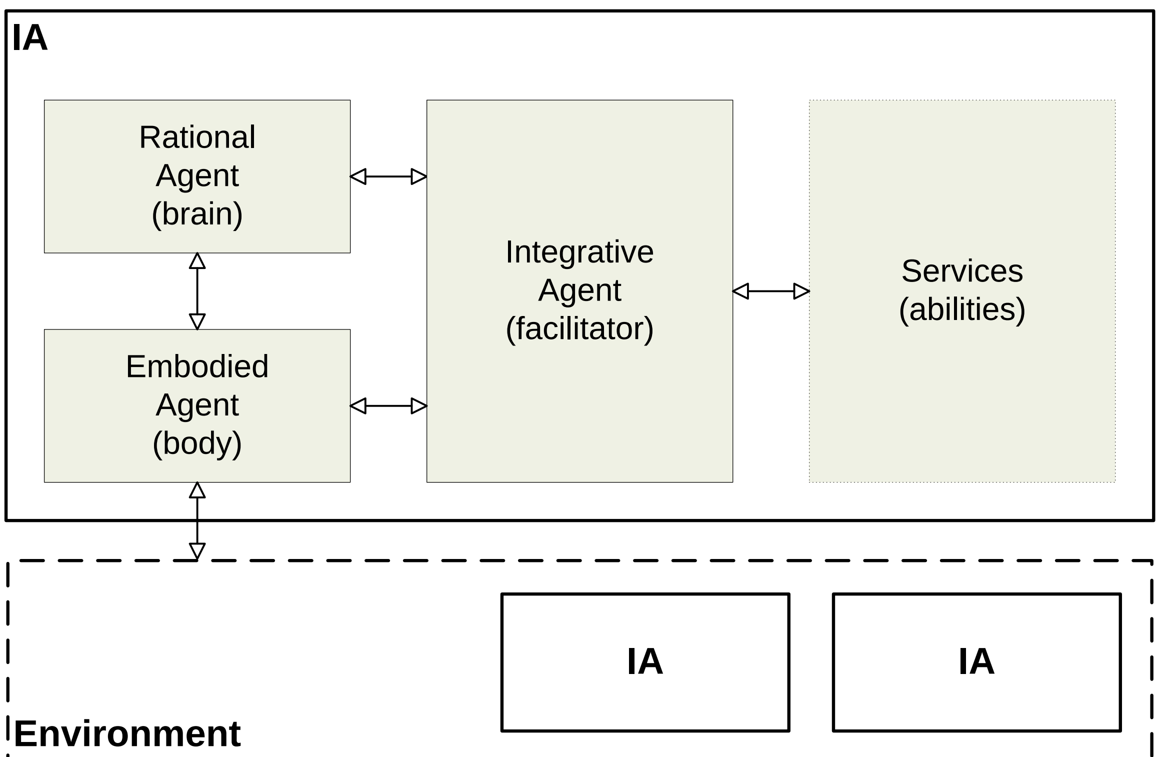

Figure 1. The SOFIA general architecture.

On the other hand, services in IA-SV are divided in two collections, those related to the environment interaction (such as sensing and acting) and those related to some intelligent process (such as learning, reasoning, planning, and others).

The IA-EA component encloses the set of services related to the abilities of sensing from the environment and interacting with it. Some of these services are, for example, virtual vision, audition or tactile capabilities.

Another important IA-EA feature associated with this component is related to the social expressions or behaviour, as for example facial expressions.

As it was mentioned previously, the internal structure of the IA-EA component has been endowed with a Service Directory (EA-SD) element to register its services (see Fig. (**2**)). These services are grouped in two collections, those related to stimuli reception (from the environment) and those related to the interaction with the environment, including expressing the IA sociality. Each of this set of services is handled by means of a corresponding element: Sensors Handled (EA-SH) and Effectors Handled (EA-EH). A fourth and last element called Integration Handled (EA-IH) allows the integration with the IA-FA to be possible, so the stimulus coming from the services could be handled by EA-SH and the actions indicated by the EA-EH could be executed by the corresponding services.

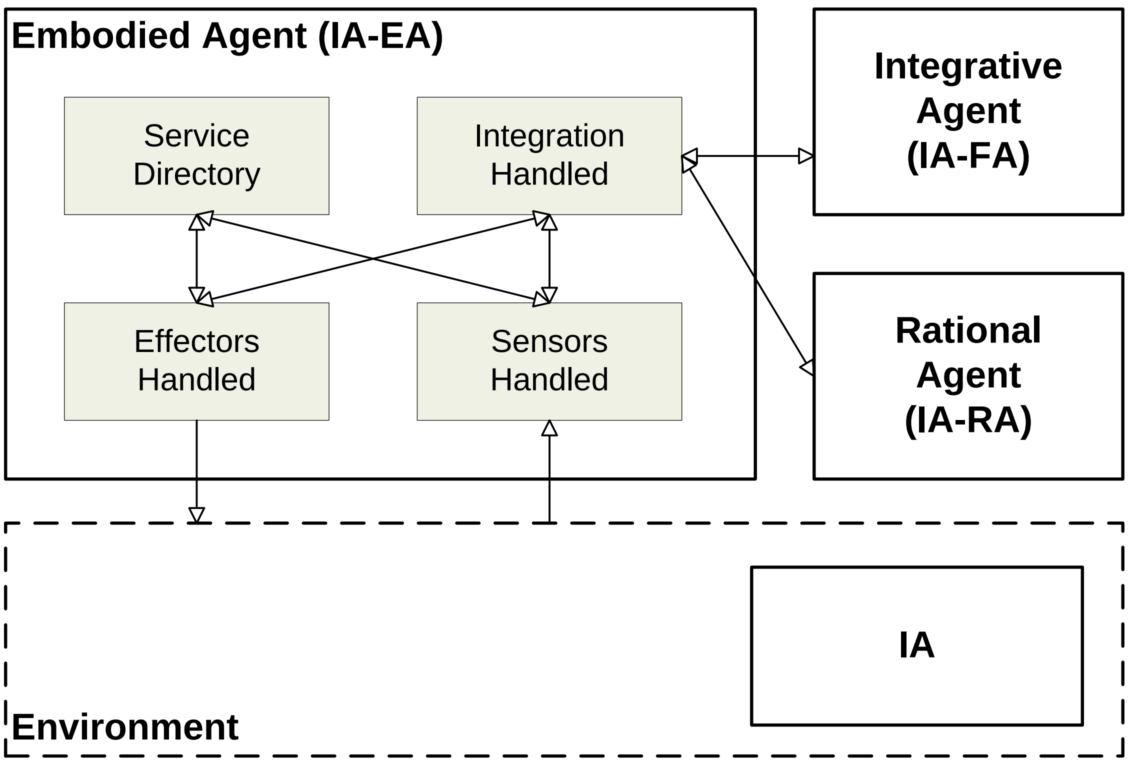

Figure 2. The IA-EA internal structure.

In the same order of ideas, the basic function of the IA-FA is to coordinate the integration/interaction between the IA-SV and the rest of the IA components. This interaction is needed when a new service is integrated with the IVA and therefore registered into the corresponding Service Directory, even when an existing service is executed. If latter, the basic idea associated to the IA-FA is to classify the services according to the ability they have.

Because of the IA-RA component, SOFIA has rational abilities. This component represents the agent's intelligent part and therefore, it encloses the set of services used for the agent to implement the process associated with these abilities. As it can be seen in Fig. (**3**), the IA-RA is a Beliefs-Desires-Intentions (BDI)-based structure [22] due to not only the fact that the BDI architecture has proved to be a useful abstraction to model autonomous agents but also to the fact that its components (beliefs, desires, and intentions) offer a convenient and intuitive way to structure the agent's action selection [23].

Due to the IA-RA component, SOFIA includes the rational and knowledgeable management abilities as part of its design, even though these capabilities can be given by the IA-SV services. Therefore, to offer this rational aspect, the IA-RA is embedded with these two elements: the Knowledge Repository (RA-KR) and the Knowledge Handled (RA-KH). Following the BDI reasoning engine model [24], the RA-KR is formed by the following information elements:

1) Beliefs: statements that the IA believes to be true and are represented by using a first-order predicate logic.
2) Goals: objectives that the IA is trying to achieve at a certain moment; any of them is associated with a status indicating if the objective was or not satisfied.
3) Plans: sequences of actions or sub-goals to be performed with the intent to reach the required goals; they are represented using a Teleo-Reactive (T-R) sequence [25].
4) Rules: IF-THEN structures [26] that can conclude in goals or sub-goals based on the information or stimuli actually known.

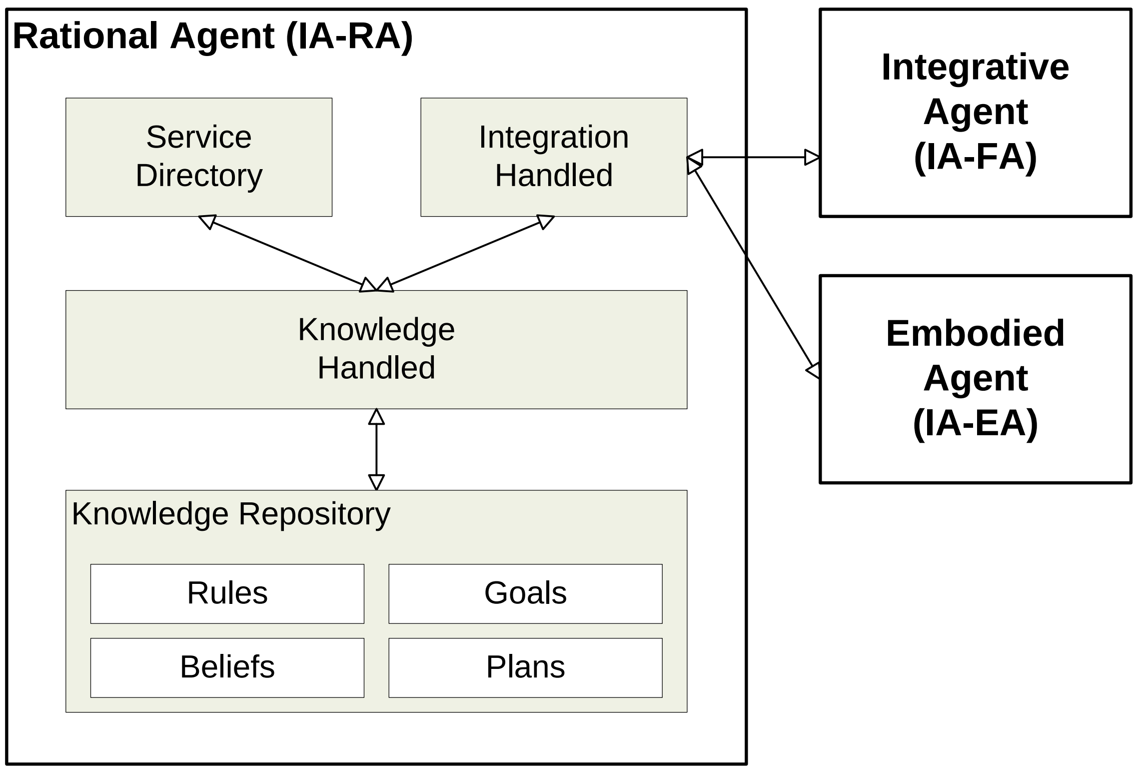

Figure 3. The IA-RA internal structure.

In the same order of ideas, the knowledge repository is represented and stored using XML based document. The XML based language used for this purpose is part of the results defined in SOFIA. The labels and attributes used to represent the information are identified properly according to the element to be represented, for example: <belief>, <goal>, <plan>, <rule>, etc.

The RA-KH encloses all the services needed not only to administrate the information that the RA-KR contains but also implement the learning and reasoning processes. Moreover the rules are used to represent the experience associated with the problem the IA is trying to resolve. The left hand side or conditional part of the rule is a combination of any of the following elements:

1) A valid goal (or sub-goal); it will have a true value if the goal status indicates that it was satisfied at the moment the rule is evaluated.
2) A valid belief statement; it will have a true value if it unifies with any of the beliefs registered in the correspondent RA-KR element.
3) A valid stimulus (see below); it will have a true value if this stimulus is recently received by the IA.

Any stimulus consists on a 2-tuple "<data, source>" that represents the information "data" coming from the environment captured by a sensor ("source"). The "source" element sets to one of the possible values defined by SOFIA and associated with a valid service category which has the ability to sense this kind of source (vision, sound, etc.). The right hand side or consequent part of the rule determines a possible reaction of the IA. It consists on a combination of the following kind of actions:

1) Set a goal state (there are two possible values: satisfied or unsatisfied).
2) Execute a specific activity (ability or service) according to the predefined categories (talking, moving, face expressing, etc.); due to the fact that it is possible to have more than one service for each category, all the services associated to this ability are executed.

The reasoning process occurs when a new stimulus arrives and the IA looks for an adequate answer. Initially, it evaluates the rules that will help to know whether there is any experience with the new information. Any rule that satisfied the conditional part is triggered (execute the actions in the consequent part) in a synchronous way, that means, changes in the goal status are not applied until all the rules are evaluated. Once all the rules have been evaluated, and if at least one rule was triggered, plans are checked first in order to review if some goals were satisfied according to the corresponding T-R sequence. After that, the rules are evaluated again and the entire process is repeated until there are no more rules.

3.2 Awareness representation and collaboration process

The collaborative process used for this research [19, 27, 28] is based on the concept of awareness by using the key awareness concepts originally proposed in [29, 30]. It has a distributed environment E containing a set of n nodes N_i ($1 \leq i \leq n$) and r different types of resources R_j ($1 \leq j \leq r$) that nodes can indifferently give. These resources can be shared as a collaborative mechanism among different nodes (such as CPU usage, memory allocation, service execution, and so on). It has:

1) $N_i.Focus(R_j)$: It can be interpreted as the subset of the space (distributed environment) on which the agent in N_i has focused his attention aiming to interact or collaborate, according to the resource R_j.

2) $N_i.NimbusState(R_j)$: Indicates the current grade of collaboration that N_i can give over R_j. It could have three possible values: *Null*, *Medium* or *Maximum*. If the current grade of collaboration N_i that is given about R_j is not high, and this node could collaborate more over this resource, then $N_i.NimbusState(R_j)$ will get the *Maximum* value. If the current grade of collaboration N_i that is given about R_j is high but N_i could improve the collaboration over this service, then $N_i.NimbusState(R_j)$ would be *Medium*. Finally, $N_i.NimbusState(R_j)$ will be *Null* if N_i cannot offer R_j or if it cannot collaborate more with this service.

3) $N_i.NimbusSpace(R_j)$: Represents the subset of the space where N_i aims to establish the collaboration about R_j.

4) $R_j.AwareInt(N_a, N_b)$: This concept quantifies the degree of collaboration over R_j between a pair of nodes N_a and N_b. It is manipulated via *Focus* and *Nimbus*, requiring a negotiation process. Following the awareness classification introduced by Greenhalgh [31], values of this concept could be *Full*, *Peripheral* or *Null*. It is calculated according to (18).

$$R_j.AwareInt(N_a,N_b)=\begin{cases} Full, & N_b \in N_a.Focus(R_j) \wedge \\ & N_a \in N_b.NimbusSpace(R_j) \\ Peripheral, & (N_b \in N_a.Focus(R_j) \wedge \\ & N_a \notin N_b.NimbusSpace(R_j)) \vee \\ & (N_b \notin N_a.Focus(R_j) \wedge \\ & N_a \in N_b.NimbusSpace(R_j)) \\ Null, & \text{other case} \end{cases} \tag{18}$$

5) $N_i.TaskResolution(R_1,\ldots,R_p)$: N_i requires collaboration with all R_j ($1 \leq j \leq p$) to solve a specific task T.

6) $N_i.CollaborativeScore(R_j)$: Determines the score to collaborate R_j in N_i. It is represented with a value within [0, 1]. The closer the value is to 0 the hardest it will be for N_i to collaborate with the necessity of R_j. The higher the value is (closer to 1) the completer will the willingness to collaborate be.

Any node N_a in the distributive environment is endowed with an agent, who has the corresponding information about E, i.e.: $N_a.Focus(R_j)$, $N_a.NimbusState(R_j)$ and $N_a.NimbusSpace(R_j)$ for each R_j. The MAS-based collaborative process in the system follows these steps (see Fig. (**4**)):

1) N_b must solve a task T by means of a collaborative task-solving process making use of the resources $R_1,\ldots,R_p$, so that, it generates a $N_b.TaskResolution(R_1,\ldots,R_p)$.

2) N_b looks for the current conditions to calculate the values associated to the key concepts of the model (*Focus*/*Nimbus* related to the other nodes), given by $N_i.Focus(R_j)$ and $N_i.Nimbus(R_j)$ $\forall i$, $1 \leq i \leq n$ and $\forall j$, $1 \leq j \leq r$. This information is used to decide the most suitable node with which to collaborate related with any resource R_j. Nodes in environment respond to request for information made by N_b. This is done through the exchange of messages between agents. As a final result of this information exchange the model will calculate the current awareness levels given by $R_j.AwareInt(N_i, N_b)$ as well as the collaboration score $N_b.CollaborativeScore(R_j)$.

3) For each resource R_j ($1 \leq j \leq p$) included in $N_b.TaskResolution(R_1,\ldots,R_p)$, N_b selects the node N_a whose $N_a.CollaborativeScore(R_j)$ is the most suitable to start the collaborative process (greatest score). Then, N_a will be the node in which N_b should collaborate on resource R_j.

4) Once N_a receives a request for cooperation, it updates its *Nimbus* (given by $N_a.NimbusState(R_j)$ and $N_a.NimbusSpace(R_j)$). In like manner, once N_a has finished collaborating with N_b it must update its *Nimbus*.

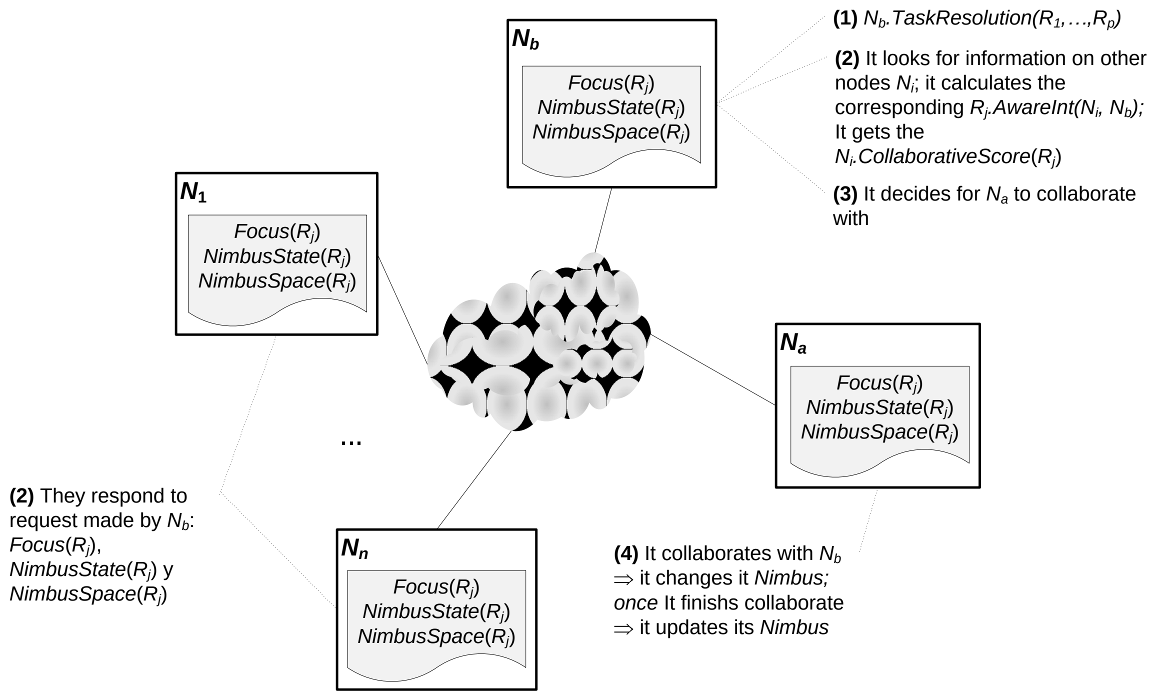

Figure 4. MAS-based collaborative process.

A SOFIA-based intelligent agent called IA-Awareness has been defined to carry out this collaboration process. Each node in the system has an IA-Awareness agent designed to take into account the following considerations/features:

1) While each node may have different agents / processes, the IA-Awareness is the one that handles and manages the collaboration process; moreover, it learns to collaborate. In this sense, any need for cooperation from some source that is currently running on the node, communicates through the IA-Awareness service $TaskResolution(R_1,\ldots,R_p)$ associated with EA-IA. This service falls under the category "Collaborate". In response to this service, IA-Awareness returns a list of p nodes, one for each resource R_j, better suited to collaborate with the current node in relation with the corresponding R_j.

2) Collaborating is one of the goals of IA-Awareness, so every time it receives a stimulus <User, $TaskResolution(R_1,\ldots,R_p)$>, the reasoning strategy of SOFIA (see Section 3.1) executes the collaboration process previously described (Fig. (**4**)).

3) There are services (abilities) associated with the embodied agent (EA-IA) that report on current levels of $Focus(R_j)$, $NimbusState(R_j)$ and $NimbusSpace(R_j)$ for a specific resource R_j. These services are associated with the category "Inform". The query for this information between IA-Awareness is achieved through the

message INFORM that is part of the communication protocol of AMBAR. The response is sent by the message PROXY.

4) Once all the necessary information is achieved, the search for the most suitable nodes to collaborate related with any R_j is done thanks to an ability associated to IA-RA and the service *FindSuitableNodes*($R_1,\ldots,R_p$) related with the category “Reason” (see Section 3.5).

5) When conditions on the environment are not appropriated enough to establish a collaboration process ($N_i.NimbusState(R_j) = Null$ for most of the N_i, R_j), the nature of the node N_b initiating a collaborative process to answer a $N_b.TaskResolution(R_1,\ldots,R_p)$ can lead to having no options, so that N_b can start a negotiation process that allows for N_b to identify new candidates to collaborate with. The detection of this saturated conditions is also an ability of IA-RA accomplished by using the service *IsOverloaded*(N, R) associated with the category “Reason” (see Section 3.5). The negotiation between IA-Awareness agents is carrying on by using a mechanism defined in AMBAR (see Section 3.3).

6) The initiation and completion of the collaboration is achieved through the implementation of services *StartCollaboration*(R) and *EndCollaboration*(R) associated with the category “Collaborate”. To communicate these actions to the agents involved in the process makes use of the corresponding messages QUERY_IF and QUERY_REF.

3.3 The negotiation mechanism

The negotiation mechanism included in AMBAR consists of three elements: 1) a heuristic algorithm used for deciding the most suitable node to initiate negotiation based on current conditions; 2) a protocol for exchanging messages between agents; 3) a heuristic method to accept/decline a need for collaboration during a negotiation.

3.3.1 Deciding the node to negotiate

For deciding the most suitable node to negotiate with, the idea is to define an unsupervised-based learning strategy aiming to correlate current information of the nodes in the distributive environment based on clusters. It is worth taking into account that “most suitable node” means a candidate that accepts the requirement established to collaborate with it, so that the negotiation is successful. Details of this strategy can be consulted in Section 3.5.4.

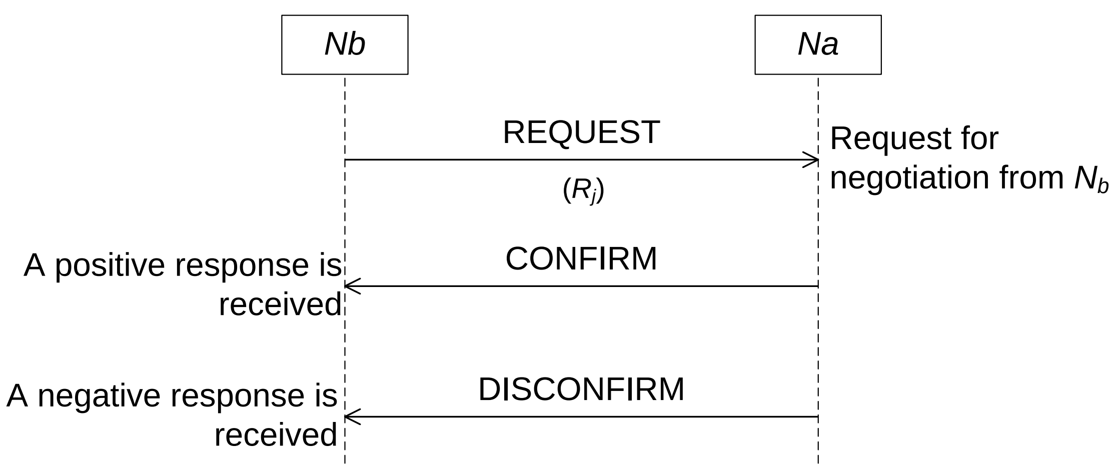

Figure 5. The inter-agent negotiation protocol.

3.3.2 The negotiation protocol

Regarding the negotiation process, agents in AMBAR exchange three possible messages (see Fig. (**5**)):

1) *REQUEST*: Once N_b has indentified a node N_a to negotiate with, N_b uses this message to communicate its needs to N_a so that N_a will accept to collaborate with N_b in relation to the resource R_j which is a parameter of the message.

2) *CONFIRM*: In response to a *REQUEST* message, N_a uses this message to inform N_b that it has accepted the request for collaboration so that N_a will change its *Nimbus*.

3) *DISCONFIRM*: In response to a *REQUEST*, N_a informs N_b that it has not accepted the request for collaboration and therefore its *Nimbus* remains unchanged.

Note that the ultimate goal of negotiation is to make a node accept to change its current condition provided by its *Nimbus*. On the other hand, in case of a negative response, N_b can decide between looking for another candidate to negotiate with and declining to seek collaboration in relation to the particular resource R_j.

3.3.3 Accept/decline collaboration

As with the decision of the most suitable node to negotiate with, this is also an ANN-based strategy. In this case there are *s* supervised ANN, one for each resource R_j in the system. All ANNs are defined in the same way (see details in Section 3.5.3).

3.4 The mutual exclusion mechanism

The nature of a node initiating a collaborative process to answer a *TaskResolution*($R_1,\ldots,R_p$), provokes a change in the conditions of the collaboration levels of the environmental nodes involved in the process. Since this information is required by the process of taking action, the levels of collaboration between the nodes turn into a critical section, so that a mutual exclusion mechanism is required. The strategy used in AMBAR is a variation of the Naimi-Tréhel's token-based algorithm [32]. In the AMBAR token-based approach [33], the token travels with a queue Q which has the nodes that require the exclusive use of the critical section and haven't been able to satisfy that need.

3.5 Heuristic-based strategies

As was mentioned before, an IA-Awareness has reasoning and learning abilities. With respect to reasoning, the agent has the following skills: 1) Find the most suitable nodes to collaborate with in response to a *TaskResolution*($R_1,\ldots,R_p$) by using the service *FindSuitableNodes*($R_1,\ldots,R_p$); 2) Detect whether or not a node N is saturated regarding a resource R by using the service *IsOverloaded*(N, R); 3) Determine whether conditions are ripe for changing the current *Nimbus* related with a R based on a need for collaboration initiated by a node N by using the service *CanChangeNimbus*(R, N); and 4) Find a suitable node to start with a negotiation process so that it changes its current *Nimbus* in relation to a resource R by using the service *FindSuitableNodeForNegotiation*(R).

3.5.1 Service *FindSuitableNodes*

It has a set of p resources R_j ($1 \le j \le p$). For each resource it must identify the most suitable node in the environment is with which to collaborate according to the corresponding resource. "Most suitable" means that it should considerer the following assumptions:

1) The node N_b that seeks collaboration should be on the *Focus* of the node N_a to identify, i.e. $N_b \in N_a.Focus(R_j)$.

2) The score of collaboration given by N_a.*CollaborativeScore*(R) must indicate the full readiness to collaborate on R (value equal or close to 1).

3) The selection must be done so that there would be a load-balancing distributed equally among all possible nodes with which to collaborate. This should take into account the current environment conditions given by $N_i.NimbusState(R_j)$ and $N_i.NimbusSpace(R_j)$ $\forall i,j$ $1 \le i \le n$, $1 \le j \le r$.

4) The answer must be given in a dynamic way (at the same time as the request is generated) and in a reasonable time.

Note that this is a load-balancing problem in distributed environments, a problem that is not new and has been approached with various techniques and methods. Strategy used to implement this service is based in SA (see Section 2.1) and originates form a pre-defined approach called SAGE (Simulated Annealing to cover dynamic load balancing in Grid Environment) [34] that later evolved into the current proposal know as CAwaSA (Collaborative Distributive Environment by means of Awareness and SA) [35].

The distributed environment is represented in CAwaSA with a set of regular $n_x n$ matrices $CDE(R_j)$ as well as a $n_x r$ matrix Nst as it is showed in (19). Rows and columns of each $CDE(R_j)$ correspond to the nodes, and each element $CDE(R_j)_{ab} = Aw_{ab}$ = {*Full*, *Peripheral*, *Null*} represents the corresponding $R_j.AwareInt(N_a, N_b)$ according to (18). On the other hand, Nst_{ij} = {*Null*, *Medium*, *Maximum*} represents the corresponding $N_i.NimbusState(R_j)$.

$$\overline{CDE(R_j)} = \begin{pmatrix} Aw_{11} & \cdots & Aw_{1n} \\ \vdots & \ddots & \vdots \\ Aw_{n1} & \cdots & Aw_{nn} \end{pmatrix}$$
$$\overline{Nst} = \begin{pmatrix} Nst_{11} & \cdots & Nst_{1r} \\ \vdots & \ddots & \vdots \\ Nst_{n1} & \cdots & Nst_{nr} \end{pmatrix} \qquad (19)$$

Related to the subject, this load-balancing problem consists on finding, in a efficient, effective and dynamical way, what $Aw_{ab}(t)$ and $Nst_{ij}(t)$ must be changed for each R_j in order to obtain new states for each $CDE(R_j)$ and Nst in presence of a $N_i.RequiredTask(R_1,\ldots,R_p)$. A solution R^b for this problem is represented with a vector of p elements r^b_l $(1 \le l \le p)$, each of them identifying the more suitable node N_a that should collaborate with N_b related with resource R_l.

To define the cost function or the energy that measures the quality of a specific solution (see Section 2.1) the following statements are considered:

1) The lower the energy is (system temperature) the better the quality in the solution will be.

2) Energy is directly proportional to the current state of environment given by all the $CDE(R_j)$ $(1 \le j \le r)$ and the Nst matrices.

3) There should be a restriction to balance the collaboration to any R_j among all the nodes in the system.

4) There should be a restriction to avoid that the node suggested to collaborate is the same node that requested the collaboration.

3.5.2 Service *IsOverload*

To know if a node N is currently saturated / overloaded in relation to the collaboration it is given over resource R, the degree of collaboration given by $N.CollaborativeScore(R)$ is used. The saturation is determined when this value is very low (0 or near 0). It makes use of the expression (20). It is important to note that this service also benefits from the learning process regarding the degrees of collaboration (see below).

$$IsOverloaded(N,R) = N.CollaborativeScore(R) < 0.125 \qquad (20)$$

3.5.3 Service *CanChangeNimbus*

This service is used to obtain a “yes” or “no” answer so that a node N_a should / can change its current conditions aiming for another node N_b to collaborate with N_a in relation with a resource R. It is supposed collaboration that can occur on R is currently saturated. This decision depends on two factors:

1) A physical aspect that is related to the current conditions of the resource and the node’s physical ability to actually be able to collaborate on the basis of this resource. For example, the maximum size of a service requests queue, the physical feature of a hardware related with the resource (CPU, memory, communications ports, and others), among others.

2) A logical aspect that deals with the relationship that N_b might had had in the past regarding a collaboration process, i.e. “to reward” those nodes that have collaborated with N_a in the past.

In this sense, what is proposed is the use of a supervised ANN (Section 2.2) that is trained based on past experience related to this situation, taking into account what was mentioned above (see details in Section 3.5.5 below).

To obtain the information related to the physical aspect associated with the resource, IA-Awareness has the ability to query the corresponding current values. This ability is implemented by using the service *GetCurrentStatus*(R) related to IA-EA and associated with the category "Inform". It returns a value between [0, 1] which expresses the percentage of current use of the node in relation to R. A value equal to 1 which means that R is being used at its maximum capacity (it is saturated).

Furthermore, each node N keeps the nxr matrix Nco whose elements Nco_{ij} matches the number of times the node N_i ($1 \le i \le n$) has collaborated with N in relation to the resource R_j ($1 \le j \le r$). This information as well as $N.Focus(R)$ are used to calculate the logical aspects needed for taking a decision.

However, it may be the case in which the ANN is not properly trained (Section 3.5.5). In this case, the decision is taken based on expression (21). The factor $\rho \in [0, 1]$ is used to determine the level of significance between the two aspects (physical and logical). *PhyAsp*(R) is determined from *GetCurrentStatus*(R) but with different interpretation, i.e. *PhyAsp*(R) = 1 indicates that the resource is completely available. The calculation of *LogAsp*(R, N) is as expressed in (22), where N_{act} is the node that should make the decision and rnd(x, y) $\in$ [x, y] is a function that returns a uniform distribution random number.

$$CanChN(R,N) = \begin{cases} \text{Yes}, & \rho PhyAsp(R) + (1-\rho) LogAsp(R,N) > 0.3 \\ \text{No}, & \text{otherwise} \end{cases} \tag{21}$$

$$\begin{aligned} & LogAsp(R,N) = Foc(R,N)(Nco_{NR}/TNco(R,N)) \\ & Foc(R,N) = \begin{cases} \text{rnd}(0.75,1), & N \in N_{act}.Focus(R) \\ \text{rnd}(0,0.25), & \text{otherwise} \end{cases} \\ & TNco(R,N) = \begin{cases} \sum_{j=1}^{s} Nco_{Nj}, & \sum_{j=1}^{s} Nco_{Nj} \neq 0 \\ \text{rnd}(Nco_{NR},1), & \text{otherwise} \end{cases} \end{aligned} \tag{22}$$

3.5.4 Service *FindSuitableNodesForNegotiation*

The objective of this service is to find / identify a node N that is a potential candidate to negotiate with, taking into account the possibility to make changes in its *Nimbus* in relation with the resource R. N can then collaborate with this node in relation with R. "Potential" means that the negotiation is successful, i.e. that N accepts.

To achieve this goal a competitive-learning-based strategy was defined aiming to correlate current information of the nodes in the distributive environment based on clusters. Therefore a NGAS-based algorithm is used. The decision consists on identifying the node closest to the hyper-plane defined by the space given by the current environment conditions. In other words, it is necessary to determine the winning unit by testing the NGAS with the environment. The goal of this learning process is 1) to cluster the input data into a set of partitions such as the intra-cluster variance which remains small compared with the inter-cluster variance, and 2) to estimate the probability density function. This clustering scheme seems possible as we expect a strong correlation among the awareness information involved (see learning details in Section 3.5.5 below).

3.5.5 Learning abilities

As mentioned in previous sections, AMBAR incorporates three ANN-based learning strategies: 1) a supervised-based method for learning collaborations based on levels of awareness; 2) an unsupervised-based method for selecting a potential candidate to negotiate on saturated conditions; and 3) a supervised-based method to learn the decision whether or not a node must change the information that describes its current conditions related with collaboration.

3.5.5.1 Learning levels of collaboration

Regarding the supervised-based ANNs for learning collaborations based on the levels of *awareness*, this process is about learning the association between the current status of the environment and the levels of collaboration obtained from that specific situation given by the $N_i.CollaborativeScore(R_j)$ ($\forall i$, $1 \le i \le n$ y $\forall j$, $1 \le j \le r$).

The ANN used in this solution, called ANN-C, has 2 inputs and 1 output. The output relates to the learned value of $N_a.CollaborativeScore(R_j)$. The inputs correspond to the following items:

1) A value $Nst \in [0,1]$ representing $N_a.NimbusState(R_j)$ that is further interpreted/ represented in (23).
2) A value $AwI \in [0,1]$ representing $R_j.AwareInt(N_a, N_b)$ that is further interpreted/ represented in (24).

$$Nst = \begin{cases} 1, & N_a.NimbusState(R_j) = Maximum \\ 0.5, & N_a.NimbusState(R_j) = Medium \\ 0, & N_a.NimbusState(R_j) = Null \end{cases} \tag{23}$$

$$AwI = \begin{cases} 1, & R_j.AwareInt(N_a, N_b) = Full \\ 0.5, & R_j.AwareInt(N_a, N_b) = Peripheral \\ 0, & R_j.AwareInt(N_a, N_b) = Null \end{cases} \tag{24}$$

To differentiate a resource from another, given the fact that each service can have a different treatment in the levels of collaboration, the IA-Awareness has a different ANN-C_j network for each resource R_j. This is an important aspect of the strategy because:

1) Each resource can be trained separately from the rest.
2) The training process is less complex and, therefore, it is expected to obtain a higher quality in the response given by each ANN-C_j.
3) The model is expansible because new ANNs can be added when a new resource has to be incorporated into the environment.
4) Each node has a particular set of ANN-C_j, i.e. IA-Awareness of each node trains and uses this ANNs according to the particular treatment a node wants to give to each resource R_j, making the collaboration model more flexible.

Since each ANN-C_j has two inputs each of them with three possible values, there is a total of nine possible patterns for training by combining the Nst values (0, 0.5, and 1) with the AwI values (0, 0.5, and 1). Moreover the strategy is implemented by using the MLP and RBFN models (see Section 2.2). The basic idea is to train both ANNs and each time the ANNs are consulted about a specific situation, one of them will choose from the two possible responses taking that which originates from the ANN that has achieved a minor error in the training process. The ANN-C_j training is performed automatically after new patterns have been stored on IA-RA. This is done by using the service $LearnAwarenessANN(R_j)$ associated to the ability "Learn".

3.5.5.2 Learning saturated conditions

This learning strategy has been identified as CAwANN (Collaborative Distributive Environment by means of an Awareness & ANN Model) which first results can be consulted in [36]. In AMBAR the NGAS-based ANN used is identified as ANN-G. The input vector is defined as follows (being N_b the node who requires collaboration on a set of services and therefore who sends the $N_b.TaskResolution(R_1,\ldots,R_p)$, for each $N_a \neq N_b$):

1) The $N_a.NimbusState(R_j)$ that will be represented by a value Nst within the interval [0,1] being $Nst = 1$ the value associated to $N_a.NimbusState(R_j) = Maximum$, $Nst = 0.5$ the value associated to $N_a.NimbusState(R_j) = Medium$, and $Nst = 0$ the value associated to $N_a.NimbusState(S_j) = Null$.
2) The $R_j.AwareInt(N_a, N_b)$ that will be represented by a value AwI within the interval [0,1] being $AwI = 1$ the value associated to $R_j.AwareInt(N_a, N_b) = Full$, $AwI = 0.5$ the value associated to $R_j.AwareInt(N_a, N_b) = Peripheral$, and $AwI = 0$ the value associated to $R_j.AwareInt(N_a, N_b) = Null$.

Therefore, the *code-vectors* for this problem have $2n$ elements, being n the number of nodes in the environment. If $N_a = N_b$ then $Nst = AwI = 0$. Patterns for learning are obtained either by those scenarios that have been stored during the dynamics of the distributed environment, or by an automatic generation, mostly random.

3.5.5.3 Learning the decision to alter the current condition

The latter case of ANN-based learning strategy is similar to that used with the ANN-C_j. In this case it has one ANN-D_j for each resource R_j. There are three inputs and one output. The output $s \in [0, 1]$ represents the decision i.e. it is accepted if $s \geq 0.5$, and declined otherwise. Inputs are as follows:

1) A value $PhyAsp(S_j) \in [0, 1]$ that indicates the level of physical availability of the resource R_j.

2) A value equal to 1 if $N_b \in N_a.Focus(R_j)$, being N_b the node that requires the decision, and N_a the node that should make the decision. If $N_b \notin N_a.Focus(R_j)$ then the entry is 0.

3) A value equal to Nco_{NR} / $TNco(R, N)$, being Nco_{NR} the number of times a node N (node that requires the decision) has collaborated with the current node (node that should make the decision) related to resource R. The idea is to reward those nodes N_b who collaborated in the past with N_a and are now requiring collaboration with N_a. $TNco(R, N)$ is calculated by following (25).

$$TNco(R,N) = \begin{cases} \sum_{j=1}^{r} Nco_{Nj}, & \sum_{j=1}^{r} Nco_{Nj} \neq 0 \\ \text{random}(Nco_{NR}, 1), & \text{otherwise} \end{cases} \quad (25)$$

The training of the ANN-D_j is performed automatically after a new pattern has been stored on IA-RA. This is done by using the service *LearnDecisionANN*(R_j) associated with the ability "Learn". An ANN-D_j is considered trained when a properly number of patterns has been stored and used. As happens with the ANN-C_j, the ANN-D_j are trained and used by using both MLPs and RBFNs strategies.

4 Implementation and evaluation

This section presents how the IA-Awareness agents were implemented as well as an explanation of the experiments conducted to test the AMBAR model.

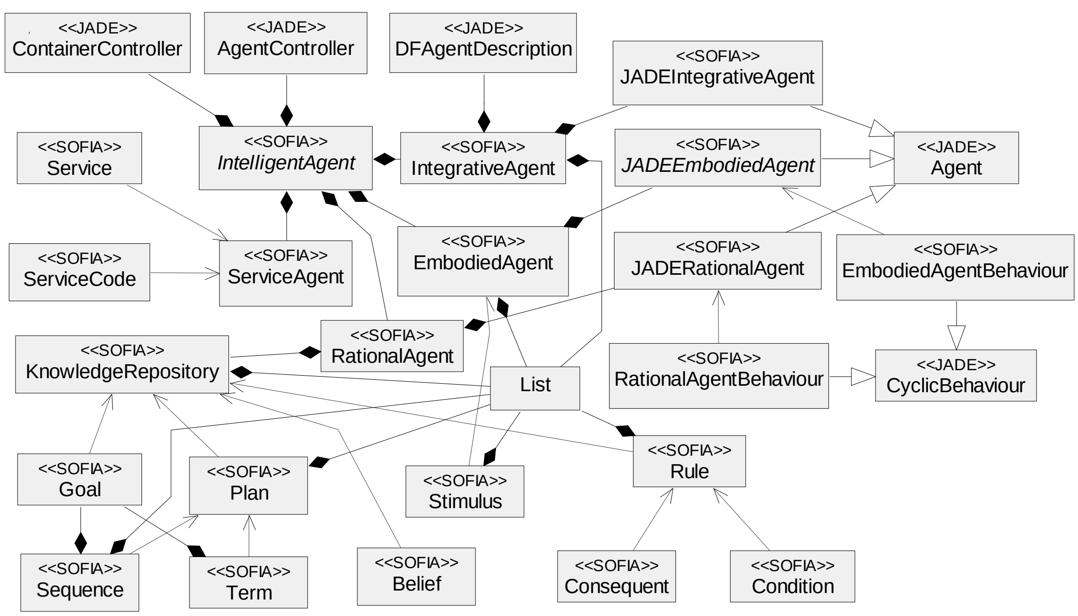

Figure 6. Class diagram of SOFIA.

4.1 Implementation

The SOFIA-based agents IA-Awareness were implemented in JADE [37] because it is FIPA-compliant as well as an open-source (based on Java). Fig. (**6**) shows the class diagram of SOFIA (identified with the stereotype "<<SOFIA>>") and its relation to classes of JADE used (identified with the stereotype "<<JADE>>"). Table 1 below has a brief explanation of each class.

Table 1: Classes for implementing SOFIA.

Class	Description
Integrative Agent	It is a specialization of the class JADEAgent used for implementing the integrator / facilitator agent or IA-FA.
JADEIntegrativeAgent	Contains the necessary elements to define the JADE-based IntegrativeAgent agent.
IntegrativeAgentBehaviour	Used for modeling the behavior of integrator agent following JADE specifications.
Embodied Agent	Used for implementing the embodied agent or IA-EA. It is a specialization of the class JADEAgent.
JADEEmbodiedAgent	Contains the necessary elements to define the JADE-based EmbodiedAgent agent.
EmbodiedAgentBehaviour	Used for modeling the behavior of embodied agent following JADE specifications.
Rational Agent	Used for implementing the rational agent or IA-RA. It is a specialization of the class JADEAgent.
JADERational Agent	Contains the necessary elements to define the JADE-based RationalAgent agent.
RationalAgentBehaviour	Used for modeling the behavior of rational agent following JADE specifications.
ServiceAgent	Used for defining and managing the services and abilities. Represents the IA-SV component.
Intelligent Agent	It is the abstraction of a SOFIA-based intelligent agent. It also defines and manages the main container required by JADE.
Service	Used for representing and managing a service, including its code.
ServiceCode	Abstraction (by using a Java interface) used for representing the code of a particular service.
Stimulus	Used for representing a stimulus.
Knowledge Repository	Allows to represent and manage the knowledge repository that is part of the IA-RA component, including its four elements: beliefs, rules, objectives and plans.
Belief	Used for representing a belief based on logic of predicates.
Goal	Used for representing a goal.
Term	Used for defining a term in a sequence of the plan. It is based on an objective.
Sequence	Used for representing a Teleo-Reactive-based sequence by using a series of terms.
Plan	Used for defining and managing plans based on sequences.
Rule	Used for representing a rule.
Condition	Used for representing a rule condition.
Consequent	Used for representing a rule, consequence or action.

The IA-Awareness agent was implemented by using the SOFIA abstract class ItelligentAgent. Fig. (**7**) shows the class diagram related with IA-Awareness and Table 2 has a brief explanation of these classes.

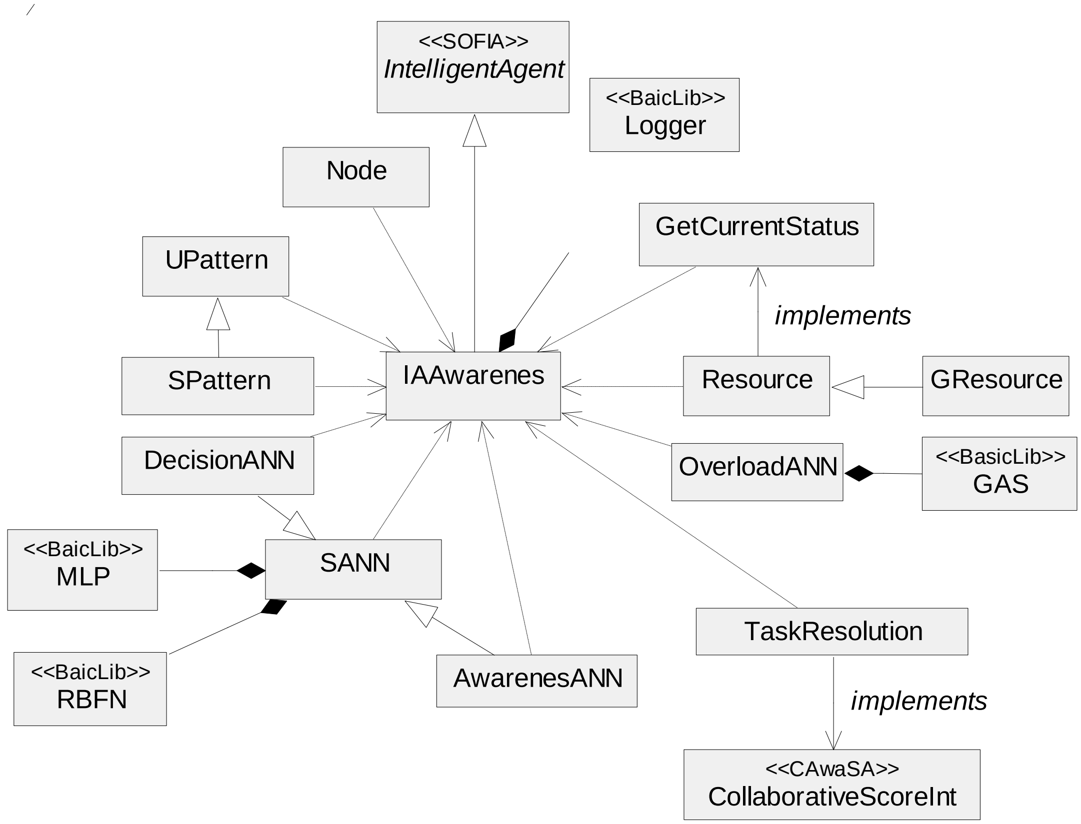

Figure 7. Class diagram related to IA-Awareness.

Table 2: Classes for implementing IA-Awareness.

Class	Description
Node	Identifies a node within the context of the distributed environment.
GResource	Define a generic resource in the environment. Generic refers to not having a specific implementation for the current state of the resource, from the physical point of view.
Resource	It is a specialization of GResource which implements an expression for the current state of the resource from the physical point of view.
GetCurrent State	Interface that allows specifying the expression to determine the current status of the resource from the physical point of view.
SANN	Generalization of a supervised-based problem by using ANN, specifically based on MLP and RBFN.
Decision ANN	Used for implementing the learning-based strategy for taking the decision to alter the current state of the node. It allows the instantiation of the ANN-D_j (Section 3.5.5.3).
Awareness ANN	Used for implementing the learning-based strategy for the awareness levels. It allows the instantiation of the ANN-C_j (Section 3.5.5.1).
Overload ANN	Used for implementing the learning-based strategy for saturated conditions. It allows the instantiation of the ANN-G (Section 3.5.5.2).
UPattern	Used for handling patterns for unsupervised-based problems (only one input vector).
SPattern	Used for handling patterns for supervised-based problems (the input-output vectors).
TaskResolution	Used for solving the load-balancing problem based on the SA algorithm.
IA-Awareness	Used for implementing the IA-Awareness.

Concerning the SOFIA knowledge repository, Table **3** shows the goals used in the IA-Awareness as well as Table **4** has the rules required by this agent. Regarding to this, for describing the conditional and consequent parts of the rules the nomenclature used in the table is: “St” for a stimulus, “Ob” for a goal, and “Sr” for a service.

Table 3: Goals used for the IA-Awareness.

Goal	Description
HasToken	Whether or not the agent has the token for the mutual exclusion algorithm.
TokenRequest	Whether or not there is a request for the token.
TaskResolution	The agent has received the stimulus <User, *TaskResolution*($R_1,\ldots,R_p$)>.
Collaborating	Whether or not the agent is currently in a collaboration process.
FindSuitable Nodes	Whether or not the agent is currently identifying the most suitable nodes to collaborate with, including the possible negotiation process.

Table 4: Rules used for the IA-Awareness.

Id	Conditionals	Consequents
1	St: <User, *TaskResolution*($R_1,\ldots,R_p$)> Ob: ¬TaskResolution	Sr: *TaskResolution*($R_1,\ldots,R_p$) Ob: TaskResolution
2	St: <User, *TaskResolution*($R_1,\ldots,R_p$)> Ob: TaskResolution	Sr: *TaskResolutionOut*($R_1,\ldots,R_p$)
3	Ob: HasToken Ob: TaskResolution Ob: ¬Collaborating	Ob: ¬TaskResolution Sr: *InitiateCollaboration*($R_1,\ldots,R_p$)
4	Ob: FindSuitableNodes	Ob: ¬FindSuitableNodes Sr: *FindSuitableNodes*($R_1,\ldots,R_p$)
5	St: <Message, QUERY_IF>	Sr: *StartCollaboration*(*R*)
6	St: <Message, QUERY_REF>	Sr: *EndCollaboration*(*R*)
7	St: <User, *InformEndCollaboration*>	Sr: *InformEndCollaboration*( )
8	St: <Message, INFORM_REF>	Sr: *SetToken*(*Q*) Ob: HasToken
9	St: <Message, INFORM_IF> Ob: HasToken	Sr: *GetToken*( ) Ob: TokenRequest
10	Ob: HasToken Ob: ¬Collaborating Ob: TokenRequest	Ob: ¬TokenRequest Sr: *SendToken*( )
11	St: <Message, AGREE>	Sr: *AckToken*( )
12	St: <Message, INFORM>	Sr: *ProcessingInfReq*(*R*, *L*)
13	St: <Message, PROXY>	Sr: *ProcessingInfRec*(*Focus*, *Nimbus*)
14	St: <Collaboration, *Negotiation*>	Sr: *MakeNegotiation*(*R*, *N*)
15	St: <Message, REQUEST>	Sr: *ProcessingRq*(*R*, *N*)
16	St: <Message, CONFIRM>	Sr: *ProcessingAns*(*R*, *N*, Yes)
17	St: <Message, DISCONFIRM>	Sr: *ProcessingAns*(*R*, *N*, No)
18	St: <Learning, Awareness>	Sr: *LearnAwarenessANN*(*R*)
19	St: <Learning, Decision>	Sr: *LearnDecisionANN*(*R*)
20	St: <Learning, Overload>	Sr: *LearnOverloadANN*()

4.2 Experimental evaluation

The evaluation of AMBAR was conducted in a TCP/IP-based LAN (Local Area Network) which assumes that each node (PC) can directly communicate with any other node. The experimentation was conducted by simulating different scenarios aiming to rate the capability of the method used for managing the growth of the nodes in the different environment conditions. The scenarios were defined by changing the quantity of nodes/PCs n (agents) as well as the number of resources r according to $n \in \{4, 8\}$ and $r \in \{2, 6, 10\}$. Therefore 6 different scenarios were simulated: 1) $n = 4$, $r = 2$; 2) $n = 4$, $r = 6$; 3) $n = 4$, $r = 10$; 4) $n = 8$, $r = 2$; 5) $n = 8$, $r = 6$; and 6) $n = 8$, $r = 10$. Moreover:

1) The initial condition of the distributed environment for each scenario (N_i.*Focus*(R_j), N_i.*NimbusState*(R_j) and N_i.*NimbusSpace*(R_j); $1 \le i \le n$; $1 \le j \le r$) was randomly defined by considering the following: one node belongs to the *Focus* of another node with a probability of 0.75 and to the *Nimbus* with a probability of 0.85.

2) All N_b nodes execute an automatic process that generates $N_b.TaskResolution(R_1,\ldots,R_p)$ by randomly selecting the involved resources from the 50% of the total resources in the scenario.

3) $PhyAsp(R_j)$, $\forall j\ 1 \leq j \leq r$ were randomly initialized.

4) The parameters used for configuring the NGAS-based ANNs are the following: $\varepsilon(0) = 1.58$; $\varepsilon(T) = 0.02$; $\rho(0) = 5.59$; $\rho(T) = 0.07$.

5) $\alpha = 0.3$ in expression (21); $\mu = 0.85$ and $\delta = 0.95$ in expressions (22) and (23) respectively; and $\rho = 0.6$ in (26).

6) Timeouts required in the model were set as follows: one minute per node for waiting the information of the current state of each node and two minutes for negotiation each service under saturated conditions.

Aiming to measure the effectiveness (θ) and efficiency (ξ) of the negotiation mechanism, expressions, (26) and (27) were defined respectively (note that both measures (θ, ξ) are positive values in [0, 1] where 1 is the maximum effectiveness and efficiency). Where:

- *PSN*: is the percentage of successful negotiations made in saturated conditions, based on the number of negotiations that receive a positive response from a node requesting to change its current saturated conditions in relation to the total attempts made.
- *MDN*: is the mean duration in seconds of the negotiation process under saturated conditions. The process starts at the moment the node requires the cooperation until it receives an answer, whether affirmative or negative. For both answers the possible retries to be made are taken into account.
- *ATC*: is the average time of collaboration in seconds calculated since $TaskResolution(R_1,\ldots,r_p)$ starts until it ends.
- *PSC*: is the percentage of successful collaborations based on the number of services in which there was positive response from a node to collaborate with, in relation to the total quantity of resources in which collaboration was required.
- *TOT*: is the number of times in which any timeout expires.
- *AMT*: is the average number of messages sent in the request for the token.
- *ATT*: is the average waiting time in seconds for the token.
- *ATB*: is the mean duration in seconds of CAwaSA method in solving the load-balancing problem.
- *ATL*: is the mean duration in seconds of the learning process.
- And: MOD is the entire model; MEM is the mutual exclusion mechanism; LBP is the load-balancing problem; NEG is the negotiation mechanism; and LPR is the learning process.

$$\begin{aligned}
&\theta(\mathrm{MOD}) = \alpha\theta(\mathrm{MEM}) + \beta\theta(\mathrm{LPB}) + \chi\theta(\mathrm{NEG}) + \delta\theta(\mathrm{LPR})\\
&\theta(\mathrm{MEM}) = 3/AMT\\
&\theta(\mathrm{LPB}) = PSC/100\\
&\theta(\mathrm{NEG}) = PSN/100\\
&\theta(\mathrm{LPR}) = (PSC + PSN)/200
\end{aligned} \tag{26}$$

$$\begin{aligned}
&\xi(\mathrm{MOD}) = \alpha\xi(\mathrm{MEM}) + \beta\xi(\mathrm{LPB}) + \chi\xi(\mathrm{NEG}) + \delta\xi(\mathrm{LPR}) - 0.05 * TOT\\
&\xi(\mathrm{MEM}) = \begin{cases} ATT/ATC, & \mathrm{A}TT <= ATC \\ ATC/ATT, & \mathrm{A}TT > ATC \end{cases}\\
&\xi(\mathrm{LPB}) = 1 - ATB/ATC\\
&\xi(\mathrm{NEG}) = 1 - MDN/ATC\\
&\xi(\mathrm{LPR}) = 1 - ATL/ATC
\end{aligned} \tag{27}$$

The factors α, β, χ and δ, all of them in [0, 1] whose sum is exactly equal to 1, determine the importance of each component of the model (MEM, LPB, NEG and LPR) for measuring θ and ξ. By setting α = 0.1, β = 0.25, χ = 0.3 and δ = 0.35 (so that gives more importance to the learning process), Table 5 shows the measures obtained after a simulation of 120 minutes for each scenario, and Fig. (**8**) shows the effectiveness and efficiency related with these measures. According to these results it is possible to make the following observations and/or conclusions:

1) The average effectiveness is 0.86 and the average efficiency is 0.88.

2) Both effectiveness and efficiency have a similar trend of behavior.

3) Nor the variation in the number of nodes or the variations in the number of resources have a particular tendency to improve or worsen the effectiveness and efficiency.

4) Without taking into account the waiting time for the token related with the MEM and its consequences over the behavior of the model, in general, the effectiveness and efficiency tends to improve with the time and therefore the learning process becomes better.

Table 5: Measures obtained from experimentation

Measure	*n*=4 *r*=2	*n*=4 *r*=6	*n*=4 *r*=10	*n*=8 *r*=2	*n*=8 *r*=6	*n*=8 *r*=10
PSN	100.00	68.13	64.29	93.75	78.13	100.00
MDN	0.00	2.14	1.23	0.13	2.16	0.44
ATC	3.40	3.46	3.34	7.47	14.87	19.28
PSC	98.96	94.24	97.70	99.89	78.08	57.90
TOT	0.00	0.00	0.00	0.00	0.00	0.00
AMT	3.05	3.01	3.02	3.04	3.28	3.64
ATT	2.47	5.86	9.58	6.73	167.38	445.15
ATB	0.00	0.02	0.05	0.06	0.68	2.03
ATL	0.01	0.02	0.02	0.01	0.03	0.02
θ(MEM)	0.98	1.00	0.99	0.99	0.92	0.83
θ(LPB)	0.99	0.94	0.98	1.00	0.78	0.58
θ(NEG)	1.00	0.68	0.64	0.94	0.78	1.00
θ(LPR)	0.99	0.81	0.81	0.97	0.78	0.79
θ(MOD)	0.99	0.82	0.82	0.97	0.79	0.80
ξ(MEM)	0.72	0.60	0.35	0.89	0.13	0.05
ξ(LPB)	1.00	0.99	0.98	0.99	0.95	0.89
ξ(NEG)	1.00	0.38	0.64	0.98	0.85	0.98
ξ(LPR)	1.00	0.99	0.99	1.00	1.00	1.00
ξ(MOD)	0.97	0.77	0.82	0.98	0.86	0.87

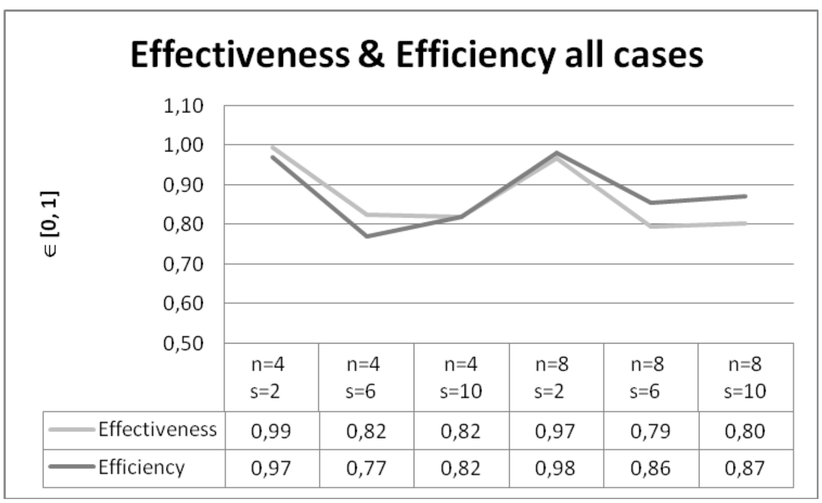

Figure 8. Effectiveness and efficiency obtained from experimentation.

It is important to stress that, due to the fact that it is a learning-based mechanism from past situations, it is assumed that as there is much more to learn, the metrics associated with it must be improved.

4.3 Functional validation

In order to validate the performance of AMBAR in real applications this section presents a possible explanation for the behavior and the application of the essential elements of the model once this is applied in the following practical scenarios:

1) Prediction of banking fraud: it represents a distribute system in which financial and national security institutions intervene with the purpose of detecting possible bank frauds during a financial transaction.
2) Design of packages of combined trips: Proposed to attend a reservation necessity that is made implicit in a trip and also to attend all the possible items involved, such as transport, room and board, and tourism. This case is related to a distributed system that possesses nodes that collaborate with each other to offer a combined package that satisfies the necessities of the travel.

4.3.1 Scenario 1: Prediction of banking fraud

The main objective of this application is to uncover bank frauds before these occur. This process benefits certain financial institutions, and their respective clients, that are related to transactions that involve money withdrawals and can be susceptible to fraud. For example, when cashing a check inside a bank and, assuming this check is fraudulent or the information presented, included the signature, is false or not valid, if the fraud goes undetected before the cashing of the check then there will be a loss of money that can put the financial institution and some of its clients in jeopardy.

According to the experience suffered by financial institutions located in countries where this type of fraud is common, the majority of these fraudulent cases are carried out by the same group of thieves. If this is true then it is possible to try to uncover the fraud through the identification of people that has committed this type of fraud before. It is also known that in the majority of cases thieves won't come back to the same financial institution that was robbed so as not to be identified for the employees that work in the institution or for the surveillance systems employed.

In the same order of ideas, recently people involved in frauds is identified automatically by the use of biometrical techniques and the existence of data bases that associate this person with any other biometrical structure like finger prints, facial features and signature just to name a few. These data bases can be used from public offices of national security (such as police departments and other departments related) and financial institutions that have adopted biometrical techniques as part of their technological solution for banking fraud. By using biometrical techniques this institutions hold biometrical information of all of the financial institution's clients and of the people that, at one time, committed fraud in the institution. With this in mind the following inquires are presented:

- How to have online access to the data bases of national public offices such as police departments?
- How can different financial institutions that cannot stand the biometric technology benefit themselves?
- How different financial institutions can share data bases so that thieves that are registered and identified in a certain society can be easily identified trying to commit the same crime in another society?

To solve these questions and to satisfy the main objective of the problem previously described a distributive collaborative environment is developed designed to be used by financial institutions and national public offices. For this it is possible to use AMBAR. The collaborative distributive environment E is formed by the nodes N_i that represent the following items:

1) Financial institutions that use biometric technology.
2) Societies that do not possess biometric technology.
3) National and State Public Offices that are in possession of data bases of current thieves.
4) Public institutions that do not manage biometric technology but require it as an activity to satisfy particular objectives.

The resources R_j are equivalent to their own purposes for the usage of the biometrical technology (enroll, identify and certify) that are associated to each one of the technologies (finger print, face and signature). Related to this the next 9 resources are encountered:

- R_1, R_2, R_3: enroll, identify and certify a finger print respectively.
- R_4, R_5, R_6: enroll, identify and certify a face respectively.
- R_7, R_8, R_9: enroll, identify and certify a signature respectively.

It is interesting to mention that if another biometric metric, different from finger print, face or signature, existed it is only necessary to add to the collaborative distributive environment new resources that are able to support the different abilities for enrolling, identifying and certifying the new metrics.

Within the same order of ideas, when it is said that $N_3.Focus(R_2) = \{N_1, N_4\}$ this is really indicating that when the system associated to N_3 has the necessity of identifying a finger print (R_2), N_3 could then ask N_1 and N_4 for collaboration. N_1 and N_4 are supposed to be nodes that possess data bases of finger prints and the abilities to achieve R_2. The expression $N_4.NimbusState(R_2) = Medium$ indicates, not only that N_4 possesses the ability to make R_2, but also, in this specific moment, N_4 is already collaborating in the making of this ability and is able to make it for longer. Speaking of $N_4.NimbusSpace(R_2) = \{N_3\}$ it is essential to make clear that in this specific moment N_4 is working on a process of identification of finger prints (R_2) due to a request of collaboration made by N_3.

4.3.2 Scenario 2: Design of packages of combined trips

This application is developed searching to answer a package of combined trips made for diverse purposes (tourism, business, pleasure and others). Businesses that work with the traveling area are benefited from this application. Generally, a package of combined travels involves a series of items such as: room and board, hotel, transportation, vehicle rental, tickets to shows or tourist attractions among others. Occasionally, clients that chose this type of businesses to request a traveling package are offered an incomplete package because the business finds it impossible to include some of the items in the package. As a consequence the client must work on some of the remaining arrangements separately from the business.

It is necessary then an application that allows businesses that work with the traveling area to collaborate with each other to complete the traveling packages. If this happened then a business that could not complete a package because it cannot offer a certain item can then complete the package by asking the missing item to some other businesses that can offer it. Businesses benefit from this because their profits rise by participating as a whole in combined packages that are requested in any of them and their clients receive a complete package without having to work on some of the missing items individually.

The nodes N_i of the collaborative distributive environment E in this case are formed by those businesses that posses one or more services related to the elements that can be a part of a combined traveling package and that wish to participate in this collaboration system. The quantity of nodes allowed to be in the system at the beginning is unlimited. The resources R_j are related in this case to the services or elements that can be a part of a combined traveling package and the related operations such as: making reservations, cancelling reservations, changing a reservation or purchase and purchase and payment of an item. For example, for an item related with transportation the following resources are at hand:

- R_1: making reservations for a certain kind of transportation.
- R_2: cancelling a reservation.
- R_3: changing a previously made reservation or purchase.
- R_4: purchasing tickets for a certain kind of transportation.

The apparition of a new element that could be a part of a combined traveling package implicates the incorporation of four new resources in the environment. These resources are associated to one of the operations previously described. The quantity of elements to be considered in this application to be a part of a combine traveling package is unlimited. Specific parameters such as the company in charge of the final service (for example: airlines, vehicle rental companies and hotels), the evaluation given from users, the number of stars of a hotel, the price and so forth, can be used to define new resources in such a way that the collaboration in this specific parameter might become centered.

Based on this representation of the environment, the interpretation that is given to expression $N_1.Focus(R_1) = \{N_2, N_3, N_4\}$, for example, is that the business represented by N_1 can, whenever there is a necessity of including in a combined traveling package the element represented by R_1 (for example make a reservation for any kind of transportation), search collaboration from the businesses N_2, N_3 and N_4 that are supposed to be habilitated to make this kind of operation. An equivalent expression to, for example, $N_1.NimbusState(R_2) = Null$ indicated that the business N_1 offers the service represented by R_2, and that N_1 is or is not collaborating in relation with the service R_2 at present. Finally, an expression equal to $N_4.NimbusSpace(R_4) = \{N_2, N_3\}$ indicates that at present N_4 is, for example, making transactions to acquire tickets for a particular transportation (R_4) due to different collaboration requests made by N_2 and N_3.

4.4 An example

There are 3 nodes in a distributed environment ($n = 3 \Rightarrow \{N_1, N_2, N_3\}$) each one is endowed with an IA-Awareness agent. There are also 4 resources with which these nodes can collaborate together ($r = 4 \Rightarrow \{R_1, R_2, R_3, R_4\}$), for example: scanning a file for viruses, temporary storage in memory, performance of a compression algorithm and search for specific information on the Internet. After a randomly initialization of the current conditions given by $N_i.\mathit{Focus}(R_j)$, $N_i.\mathit{NimbusState}(R_j)$ and $N_i.\mathit{NimbusSpace}(R_j)$ $\forall i,j$ $1 \leq i \leq 3$, $1 \leq j \leq 4$ is:

$N_1.\mathit{Focus}(R_1) = \{N_3\}$	$N_1.\mathit{Focus}(R_2) = \{N_2, N_3\}$
$N_1.\mathit{NimbusState}(R_1) = \mathit{Null}$	$N_1.\mathit{NimbusState}(R_2) = \mathit{Null}$
$N_1.\mathit{NimbusSpace}(R_1) = \{N_2\}$	$N_1.\mathit{NimbusSpace}(R_2) = \{N_2, N_3\}$
$N_1.\mathit{Focus}(R_3) = \{N_2\}$	$N_1.\mathit{Focus}(R_4) = \{N_2, N_3\}$
$N_1.\mathit{NimbusState}(R_3) = \mathit{Medium}$	$N_1.\mathit{NimbusState}(R_4) = \mathit{Null}$
$N_1.\mathit{NimbusSpace}(R_3) = \{N_2, N_3\}$	$N_1.\mathit{NimbusSpace}(R_4) = \{N_2, N_3\}$
$N_2.\mathit{Focus}(R_1) = \{N_3\}$	$N_2.\mathit{Focus}(R_2) = \{N_1, N_3\}$
$N_2.\mathit{NimbusState}(R_1) = \mathit{Maximum}$	$N_2.\mathit{NimbusState}(R_2) = \mathit{Medium}$
$N_2.\mathit{NimbusSpace}(R_1) = \{N_1, N_3\}$	$N_2.\mathit{NimbusSpace}(R_2) = \{N_1, N_3\}$
$N_2.\mathit{Focus}(R_3) = \{N_1, N_3\}$	$N_2.\mathit{Focus}(R_4) = \{N_1, N_3\}$
$N_2.\mathit{NimbusState}(R_3) = \mathit{Medium}$	$N_2.\mathit{NimbusState}(R_4) = \mathit{Null}$
$N_2.\mathit{NimbusSpace}(R_3) = \{N_1, N_3\}$	$N_2.\mathit{NimbusSpace}(R_4) = \{N_1, N_3\}$
$N_3.\mathit{Focus}(R_1) = \{N_1, N_2\}$	$N_3.\mathit{Focus}(R_2) = \{N_1, N_2\}$
$N_3.\mathit{NimbusState}(R_1) = \mathit{Medium}$	$N_3.\mathit{NimbusState}(R_2) = \mathit{Medium}$
$N_3.\mathit{NimbusSpace}(R_1) = \{N_1, N_2\}$	$N_3.\mathit{NimbusSpace}(R_2) = \{N_1, N_2\}$
$N_3.\mathit{Focus}(R_3) = \{N_1, N_2\}$	$N_3.\mathit{Focus}(R_4) = \{N_1, N_2\}$
$N_3.\mathit{NimbusState}(R_3) = \mathit{Medium}$	$N_3.\mathit{NimbusState}(R_4) = \mathit{Null}$
$N_3.\mathit{NimbusSpace}(R_3) = \{N_1, N_2\}$	$N_3.\mathit{NimbusSpace}(R_4) = \{N_2\}$

N_3 needs to execute a task that involves the resources R_1, R_3 and R_4 and for this it requires collaboration with the resources R_3 and R_4 so that it generates a $N_3.\mathit{TaskResolution}(R_3, R_4)$. Therefore, the process explained in Section 2.2 occurs.

The conditions related to R_4 are saturated (see $N_1.\mathit{NimbusState}(R_4)$ and $N_2.\mathit{NimbusState}(R_4)$) so that IA-Awareness in N_3 should execute the negotiation mechanism explained in Section 3. As a consequence, N_3 decided (by using the method) that the node to negotiate with is N_1. Further into the negotiation protocol between N_3 and N_1, N_1 was agreed to change its current conditions aiming to collaborate with N_3 in relation to R_4.

5 Related Work

In [38] there is an example of a context enhanced messaging and collaboration system in which resources related to the context of text messages may be automatically identified and exchanged between users in association with the text messages. This is an example of a collaborative distributed system in which a particular resource is exchanged. In AMBAR the collaboration could be possible with different kind of resources and not only by exchanging it but also by using it. Another example related with collaboration in distributed systems can be consulted in [39], a system which includes a collaboration manager coordinating collaboration (asynchronous or synchronous) between the nodes in the distributed system.

On the other hand, researchers in CSCW have already proposed awareness to: 1) give information on the surroundings of the target user [40]; 2) provide common or public space where users can gather and meet [41]; 3) simulate informal communicative opportunities in real world using computers [42].

Regarding the context of awareness and recognizing the current context of a user or device, authors in [43] present an approach based on general and heuristic extensions to the growing NGAS algorithm classifier which allow its direct application for context recognition. The authors here used context awareness features for automatically classifying sensor data to recognize user or device context.

Another example of using ANN in a problem related with collaboration can be consulted in [44]. In this work, authors present an architecture aiming to address problems in a collaborative learning activity to create groups among students. They used a neural network algorithm to obtain homogenous groups.

The use of ANN technology for negotiation algorithms can be found in [45-49]. Authors in [45] present an adaptive negotiation model that uses a feed-forward artificial neural network as a learning capability to model the other agent negotiation strategy. In [46], authors proposed a MLP-based learning technique that is used mainly to detect at an early stage the cases where agreements are not achievable, supporting the decision of the agents to withdraw or not from the specific negotiation thread. The basic idea is to enhance such agents with techniques enabling them to predict their opponents' negotiation behaviour and thus achieve more profitable results and better resource utilization.

In the same order of ideas, authors in [48] focus on the transmission of what an agent is thinking aiming to the emergence of the autonomous and decentralized arbitration through communication among some agents. In this approach the communication contents and protocols are not prescribed and are acquired by learning using a reinforcement signal which is given to the agent after its action. On the other hand, authors in [49] propose an agent-based learning method in automated negotiation based on ANN aiming to implement interactions between agents and guarantee the profits of the participants for reciprocity. Finally, authors in [47] overcome the difficulty of using fuzzy logic and fuzzy neural networks by applying an adaptive neural topology to model the negotiation process.

The patent referenced in [50] is related with a learning environment having a plurality of tables and a plurality of technology display devices distributed about the environment for use by occupants of the learning environment. It is an example of a learning strategy but AMBAR is more than one strategy, it is a model that includes learning techniques to collaborate in distributed environments.

Although the combination of ANN and some processes related with collaboration in distributed environments as negotiation can be found in several previous works, as far as we know, there is not any model for learning cooperation on distributed environments by using the awareness information originated in this system and used for improving future collaborations.

6 Current and Future Developments

This paper presents AMBAR a new entire collaboration model used for a multi-agent based system in collaborative distributed environments. The method proposed is endowed with heuristic algorithms and a message protocol between agents. The heuristic algorithms are used for the agents: 1) to solve the load-balancing problem, 2) to decide for the most suitable node to collaborate with, and 3) to decide whether or not to collaborate.

AMBAR has been designed for distributed environments in which its nodes are aware of the existence of others, at least those that can establish a collaborative process. Results show that AMBAR is 86% effective and 88% efficient. Therefore, this model ensures collaboration in these environments in short time.

Although this method has not yet been tested in real scenarios, it has been designed to be suitable for true distributed environments. In fact the experimentation and validation were carried out to demonstrate that this method could be extended to real scenarios with no problems.

We are currently working on:

1) Testing this method in real environments.
2) Using the model in actual end-use applications.
3) Proposing some variants with alternatives for improving the mutual exclusion mechanism.

Acknowledgements

This research was supported by the Informatics and Computer Technology Research Center (CITEC – from the Spanish "Centro de Investigación en Informática y Tecnología de la Computación") at the Universidad Nacional Experimental de Guayana (UNEG) of Venezuela.

Conflict of Interest

Authors do not have any conflict of interest.

References

1. Matsushita Y, Okada K, Eds. Collaboration and Communication; Distributed collaborative media series 3; Kyoritsu Press 1995.
2. Kuwana E, Horikawa K. Coordination Process Model-based Design for Synchronous Group Task Support System; Technical Reports of Information Processing Society of Japan; Groupware; No. 13; 1995: 1-6.
3. Malone TW, Crowston K. The interdisciplinary study of coordination: ACM Computing Surveys. ACM 1994; 26(1): 87-119.
4. Ogata H, Yano Y. Knowledge Awareness: Bridging Learners in a Collaborative Learning Environment. Inter J Educ Telecom 1998; 4(2): 219-236.
5. Gutwin C, Stark G, Greenberg S. Support for Workspace Awareness in Educational Groupware, *Proceeding of Computer Supported Collaborative Learning* (CSCL '95), 1995: 147-156.
6. Kirkpatrick S. Optimization by simulated annealing: Quantitative Studies. J Statistical Phy 1984; 34(5-6): 975-986.
7. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equations of state calculations by fast computing machines. J Chem Phy 1953;21(6): 1087-1091.
8. Kohonen T, Mäkisara K, Saramäki T. Phonotopic maps- insightful representation of phonological features for speech recognition. *Proceeding of 7th International Conference on Pattern Recognition* 1984: 182-185.
9. Makhoul J, Roucos S, Gish H. Vector quantization in speech coding. Proceeding of IEEE 73; IEEE Computer Society; 1985: 1551-1588.
10. Nasrabadi NM, Feng Y. Vector quantization of images based upon the Kohonen self-organizing feature maps. *Proceedings of IEEE International Conference on Neural Networks*; IEEE Computer Society 1988: 1101-1108.
11. Nasrabadi NM, King RA. Image coding using vector quantization: A review. IEEE Trans Comm 1988; 36(8): 957-971.
12. Naylor J, Li KP. Analysis of a Neural Network Algorithm for vector quantization of speech parameters; in Proc First Annual INNS Meeting, NY; Pergamon Press; 1988: 310-315.
13. Martinetz TM, Schulten KJ. A neural gas network learns topologies. In: Kohonen T, Mäkisara K, Simula O, Kangas J (Eds.). Artificial Neural Networks 1991: 397-402.
14. Lingireddy S, Ormsbee LE. Neural networks in optimal calibration of water distribution systems. Artificial Neural Networks for Civil Engineers: Advanced Features and Applications 1998: 53-76.
15. Shahsavand A, Ahmadpour A. Application of Optimal RBF neural networks for optimization and characterization of porous arterials. Comput Chem Engineering 2005; 29: 2134-2143.
16. Jin R, Chen W, Simpson TW. Comparative studies of metamodelling techniques under multiple modeling criteria. Struct Multidiscip Optim 2001; 23: 1-13.
17. Haykin S. Neural Networks: A Comprehensive Foundation; Prentice Hall; ISBN: 0-132-73350-1; 1998.
18. Werbos P. The roots of backpropagation: From ordered derivatives to neural networks and political forecasting; Wiley-Interscience, Adaptive and Learning Systems for Signal Processing, Communications and Control Series; ISBN: 0-471-59897-6; 1994.
19. Paletta M, Herrero P. Towards fraud detection support using grid technology. Multiagent and Grid Systems - An International Journal 5; DOI 10.3233/MGS-2009-0131; IOS Press; 2009: 311-324
20. Paletta M, Herrero P. Awareness-based learning model to improve cooperation in collaborative distributed environments. *In Proc of 3rd International KES Symposium on Agents and Multi-agents Systems Technologies and Applications (KES-AMSTA 2009*); Håkansson A *et al*. (Eds.); Springer; LNAI 5559; 2009: 793-802.
21. Foundation for Intelligent Physical Agents (2002). FIPA Abstract Architecture Specification; SC00001; Geneva, Switzerland; http://www.fipa.org/specs/ fipa00001/index.html.
22. Rao AS, Georgeff MP. Modeling rational agents within a BDI-architecture. *In Proc Second International Conference on Principles of Knowledge Representation and Reasoning;* Morgan Kaufmann publishers Inc.; 1991: 473-484.
23. Guye-Vuilleme A, Thalmann D. A high-level architecture for believable social agents. Virtual Reality (UK); Vol. 5; Num. 2; Springer; 2000: 95-106.
24. Pokahr A, Braubach L, Lamersdorf W. Jadex: A BDI Reasoning Engine; Multiagent Systems, Artificial Societies, and Simulated, Organizations International Book Series; Vol. 15; 10.1007/b137449; ISBN: 978-0-387-24568-3; Springer; 2005: 149-174.

25. Nilsson NJ. Teleo reactive programs for agent control. J Artif Intell Res 1994; 1: 139-158.
26. Post E. Formal reductions of the general combinatorial problems. American J Mathematics 1943; 65: 197-268.
27. Paletta M, Herrero P. Foreseeing cooperation behaviors in collaborative grid environments. *In Proc of 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS'09*); Springer; Vol. 50; 2009: 120-129.
28. Paletta M, Herrero P. Learning cooperation in collaborative grid environments to improve cover load balancing delivery. *In Proc of IEEE/WIC/ACM Joint Conferences on Web Intelligence and Intelligent Agent Technology* IEEE Computer Society E3496; 2008: 399-402.
29. Herrero P, Bosque JL, Pérez MS. An agents-based cooperative awareness model to cover load balancing delivery in grid environments; In Proc. On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops; Springer; LNCS 2536; Vol. 4805; 2007: 64-74.
30. Herrero P, Bosque JL, Pérez MS. Managing dynamic virtual organizations to get effective cooperation in collaborative grid environments; in Proc. On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops; Springer; LNCS 2536; Vol. 4804; 2007: 1435-1452.
31. Greenhalgh C. Large Scale Collaborative Virtual Environments. PhD Thesis, University of Nottingham, UK,1997.
32. Naimi M, Trehel M, Arnold A. A log (N) distributed mutual exclusion algorithm based on path reversal. J Parallel Distributed Comput 1996; 34(1): 1-13.
33. Paletta M, Herrero P. A Token-Based Mutual Exclusion Approach to Improve Collaboration in Distributed Environments. I*n Proc. 1st International Conference on Computational Collective Intelligence – Semantic Web, Social Networks & Multiagent Systems (ICCCI 2009*); Nguyen NT, Kowalczyk R, Chen SM (Eds.); LNAI 5796; Springer; 2009: 118-127.
34. Paletta M, Herrero P. Simulated annealing method to cover dynamic load balancing in grid environment. *In Proc of International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 08)*; Salamanca, Spain, October 22-24, 2008: 1-10.
35. Paletta M, Herrero P. An awareness-based simulated annealing method to cover dynamic load-balancing in collaborative distributed environments. In Proc. 2009 IEEE/WIC/ACM International Conference on Intelligence Agent Technology (IAT 2009); Baeza-Yates R et al (Eds.); IEEE Computer Society; 2009: 371-374.
36. Paletta M, Herrero P. An awareness-based artificial neural network for cooperative distributed environments. *In Proc International Work Conference on Artificial Neural Networks (IWANN 2009);* Cabestany J et al (Eds.); LNCS Vol. 5517; ISBN: 978-3-642-02477-1; 2009: 114-121.
37. Bellifemine F, Poggi A, Rimassa G. JADE-A FIPA-compliant agent framework; Telecom Italia internal technical report. *In Proc. International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAM'99*); 1999: 97-108.
38. Goodman, B.D., Jania, F.L., Shaw, D.M.: US2009007503007 (**2009**).
39. Morita, M., Fors, S.L., Hughes, W.D.: US20080256181A1(**2008**).
40. Dourish P, Bly S. Portholes: Supporting awareness in a Distributed Work Group. *in Proc. Conference on Human Factors in Computing Systems (CHI'92)*; 1992: 541-548.
41. Fish R, Kraut R, Chalfonte B. *The Video Window System in Informal Communications; in Proc. Computer Supported Cooperative Work (CSCW'90);* 1990: 1-12.
42. Matsuura N, Hidaka T, Okada K, Matsushita Y. VENUS: an informal communication environment supporting interest awareness. Trans Information Processing Society of Japan 1995; 36(6): 1332-1341.
43. Mayrhofer R, Radi H. Extending the Growing Neural Gas Classifier for Context Recognition; Computer Aided Systems Theory. EUROCAST 2007: 920-927.
44. Blanchard E, Frasson C. Designing a Multi-agent Architecture based on clustering for collaborative learning sessions; *In Proc International Conference in Intelligent Tutoring Systems*; LNCS. http://citeseerx.ist.psu.edu/viewdoc/ summary?doi=10.1.1.86.4799; Springer; 2002.
45. Oprea M. An Adaptive negotiation model for agent-based electronic commerce. Studies in Informatics and Control 2002; 11(3): 271-279.
46. Roussaki I, Papaioannou I, Anangostou M. Building automated negotiation strategies enhanced by MLP and GR neural networks for opponent agent behaviour prognosis. Comput Ambient Intell 2007; 4507: 152-161.

47. Sakas DP, Vlachos DS, Simos TE. Adaptive Neural Networks for Automatic Negotiation. I*n Proc. of the International Conference on Computational Methods in Science and Engineering (ICCMSE 2007);* Vol. 2; Parts A and B; AIP Conference Proceedings; 2007: 1355-1358.
48. Shibata K, Ito K. Learning of Communication for Negotiation and Emergence of Individuality by Reinforcement Learning using Neural Network; Trans. of The Society of Instrument and Control Engineers (SICE); Vol.35; No.11; 1999: 1346-1354.
49. Zeng ZM, Meng B, Zeng YY. An Adaptive learning method in automated negotiation based on artificial neural network. *In Proc of 2005 International Conference on Machine Learning and Cybernetics*; Vol. 1; Issue 18-21; 2005: 383-387.
50. Valoe, E., Barnhart, J., Barton, H., Curtis, J.A.: US20090000228A1 (**2009**).